RasPi DESIGN BUILD CODE

32

Get hands-on with your Raspberry Pi

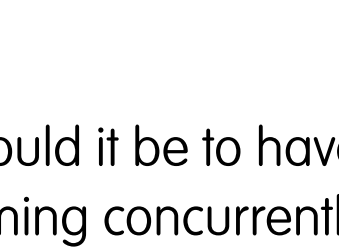
SET UP A MULTI-ROOM SOUND SYSTEM

+ HANDLE **OPENGL** ON YOUR PI

Plus Make a networked sensor display

Welcome

How cool would it be to have audio streaming concurrently in every room of your home? It's ideal for playing music at a party, but equally useful if you're simply pottering around doing the housework and don't want to miss a key part of the audio book you're listening to. Traditionally though, multi-room audio used to be an expensive setup. Not any more! The Raspberry Pi can take care of that, giving you your favourite sounds wherever you are. Find out how in our feature this issue! Also in this issue we take you through the second part of building your own Pi-based version of *Rock, Paper, Scissors, Lizard, Spock*, as well as looking at networked sensors, Open GL and more. Enjoy!



Editor

Get inspired

Discover the RasPi community's best projects

Expert advice

Got a question? Get in touch and we'll give you a hand

Easy-to-follow guides

Learn to make and code gadgets with Raspberry Pi

From the makers of

LinuxUser & Developer

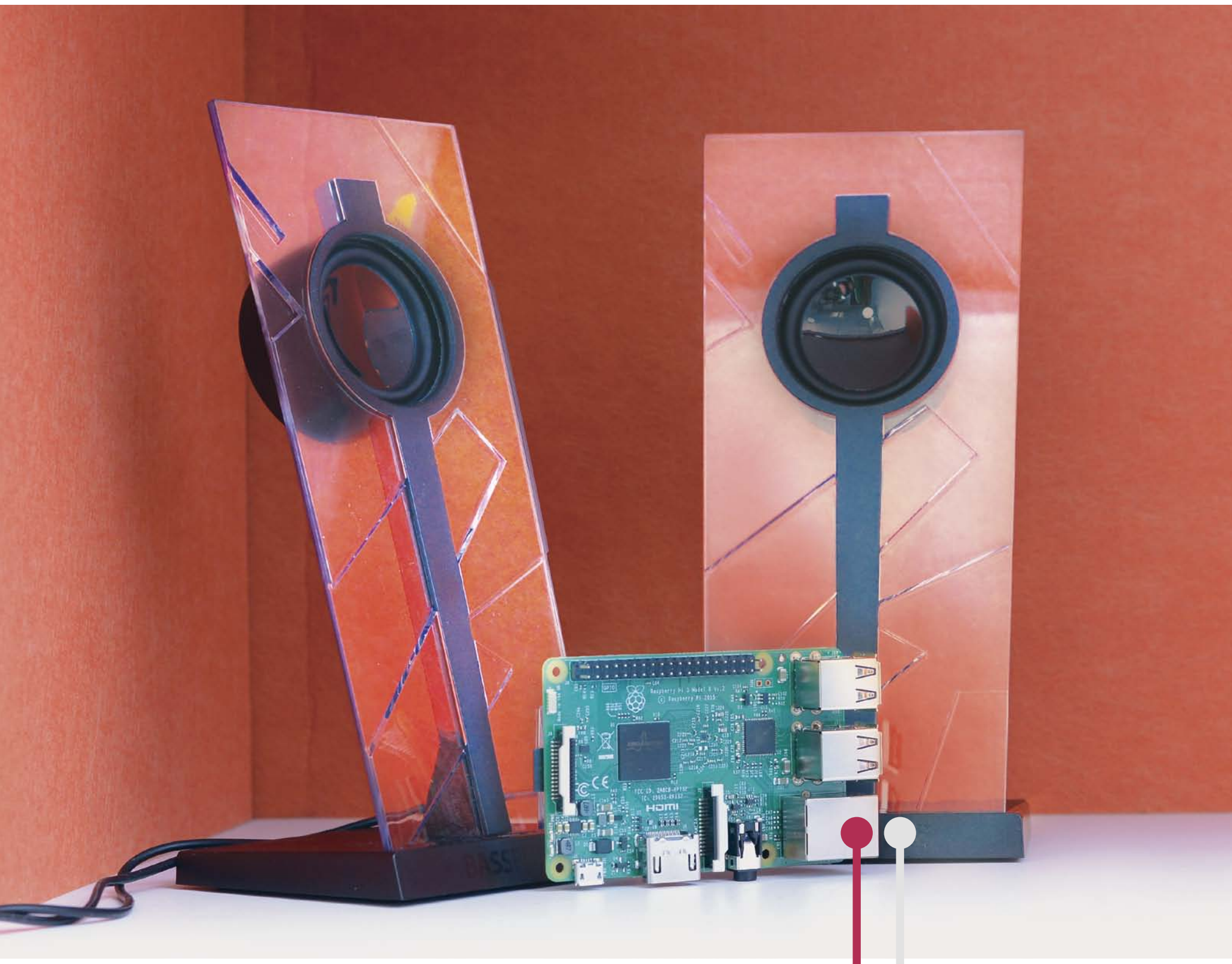
Join the conversation at...

@linuxusermag | Linux User & Developer | RasPi@imagine-publishing.co.uk

Contents

Set up a multi-room sound system

Use a Raspberry Pi running Logitech's Squeezebox software to provide synchronised multi-room audio



In this article we are going to use Logitech's Squeezebox Media Server and the squeezelite client to create a multi-room audio system. The media server can stream audio to each squeezelite client and also synchronise the playback between multiple clients if desired. Logitech open-sourced the Squeezebox software after officially discontinuing the hardware line.

The media server can be controlled from a web interface or various smartphone applications. Unfortunately, the Spotify plugin (specifically the third-party Spotify plugin by Triode) was partially broken at the time of writing. Search isn't working, likely because the Spotify API has been updated since the plugin was last updated. Instead, the server can be pointed at a music directory on the Pi. You can either put music directly onto the SD card or connect up a USB flash drive or hard drive. As you can get a 64GB flash drive for around £15 you should be able to fit plenty on there. We will set up a samba server on the music server so you can copy music to it over the network, instead of needing to disconnect the flash drive and disrupt playback.

We recommend that you use a Pi 2 for the server but an older Pi or a Pi Zero should have enough power to be a squeezelite client.

THE PROJECT ESSENTIALS

A Raspberry Pi for each room you would like to play music in (one of these can act as a server or a server can be separate)

Ideally a USB sound card for each Pi (a Behringer UCA-202 is a great option)

Speaker amplifier and speakers to plug the Pi into

A wired or wireless network

A USB Wi-Fi adapter for each Pi you want to be wireless (an Edimax EW-7811UN works out of the box)

01 Flash SD cards

We're going to be using Raspbian Jessie for this. We used the Raspbian Jessie Lite version because it is smaller and only comes with a minimal set of software. However, it doesn't really matter if you use the full version, or even the NOOBS installed version. You'll need an SD card for each Pi that you are using. In our case, this would be one for a server/client, and one client.

Above A Raspberry Pi running Logitech's Squeezebox service can be made into a homebrew media server that can play music and internet radio stations

02 Power on your Pi

It's a good idea to power on each Pi one at a time so you know which one is which. Log into your router / dhcp server to find the address of your pi, or you might just be able to SSH into it using the name "raspberrypi". Alternatively you can use nmap or "arp -na | grep -i b8:27:eb" to find every Pi on the network.

03 Logging in

For each Pi, loG in using ssh mailto:pi@172.17 .173.58"pi@172.17.173.58, replacing the IP with the IP address of your Pi. Use the password 'raspberry'. Then set the hostname to a useful name (such as livingroom, or musicserver): echo 'musicserver' | sudo tee /etc/ hostname

Then use sudo raspi-config to expand the filesystem, then select finish and reboot. After this, hopefully you'll be able to log into the Pi using the name you just set. At the very least you'll know which

one you are logged in to.

04 Wi-Fi setup

If you are using a Wi-Fi adapter then connect it up to the Pi and then edit: /etc/wpa_supplicant/wpa_supplicant.conf with sudo, and your editor of choice – for example: sudo nano /etc/wpa_supplicant/wpa_supplicant.conf – then add the following lines:

```
network={
    ssid="MYSSID"
    psk="passphrase"
}
```

If you reboot and remove the ethernet cable, the Pi should connect over Wi-Fi.

05 Sound card setup

If you are using a USB sound card then you want it to have priority over the built-in sound card (which is always loaded as card0):

```
pcm.!default {
   type hw
   card 1
}

ctl.!default {
   type hw
   card 1
}
```

Then use alsamixer to set the volume to 100% because we will use it as a line out device.

Having audio dropouts?

If you are having audio dropouts then the most likely cause is that there is not enough network bandwidth available. The easiest way to solve this is to move your Pi from Wi-Fi to a wired network. If you don't want to run network cables around the house you could try a different Wi-Fi adapter, or you could try using Powerline Ethernet adapters. You can pick a pair of these up for -£25 and then add more as you need. Commercial devices such as ▶

06 Install squeezelite on each client

Once you have set up Wi-Fi and USB Sound cards on the clients (and server if that is also acting as a client), it's time to install squeezelite:

```
apt-get update
apt-get upgrade
apt-get install squeezelite
```

That's all it takes to set up a client. The squeezelite client will start itself and everything else is controlled through the server, so now it's time to set that up.

07 Preparing music storage

We are going to store our music in /mnt/music on the media server Pi. If you want to use a USB hard drive or flash drive for additional storage space then you'll need to format it and then mount it in that directory:

```
sudo parted /dev/sda
(parted) mktable msdos
(parted) mkpart primary ext2 0% 100%
(parted) quit
sudo mkfs.ext4 /dev/sda1
sudo mkdir /mnt/music
sudo mount /dev/sda1 /mnt/music

sudo editor /etc/fstab, add:
/dev/sda1 /mnt/music ext4 defaults,noatime
0 2
```

08 Set up SAMBA share

Samba is an implementation of the file-sharing

the SONOS speakers use their own Wi-Fi network just above the standard 2.4GHz range so that the network is not congested, avoiding this problem. Using 5GHz Wi-Fi could also be an option.

protocol used by Windows. It is the best choice because it is supported by Windows, Mac and Linux. This will allow you to simply drag and drop music onto your media server from any PC.

```
sudo apt-get update
sudo apt-get install samba
```

```
sudo mv /etc/samba/smb.conf /etc/samba/smb.conf. original
sudo vim /etc/samba/smb.conf:
[global]
   workgroup = WORKGROUP
   server string = Music Server
   security = user
   guest account = nobody
   map to guest = Bad User
```

```
[music]
   path = /mnt/music
   public = yes
   browsable = yes
   only guest = yes
   writable = yes
```

```
sudo systemctl restart smbd
sudo chown nobody:nogroup /mnt/music/
```

Make /mnt/music readable, writable and executable by all.

```
sudo chmod -R 777 /mnt/music/
```

Cost of building

Assuming you already have an old Pi or two lying around, this can be a fairly cheap project – there are plenty of good quality class T speaker amps around that you can get for £20 or so. Just search 'class T amp' on Amazon and you'll see several results. These work great as long as you don't enable the tone controls, and they'll easily go loud enough that your neighbours will complain before they distort. On top of that you'll need some bookshelf speakers, a USB sound card, some speaker wire and ▶

09 Copy music to server

You can now copy music to the server. On Windows, Mac, or Linux the server should be visible in the "Network" section. On Linux using the Thunar file browser, we had to go to "Network" > "Windows Network" > "WORKGROUP" > "MUSICSERVER". Then we copy and paste the music we want over to the server. Most file formats should be used but we used 320k MP3 files, all tagged correctly to make sure they display properly once loaded into Logitech Media Server.

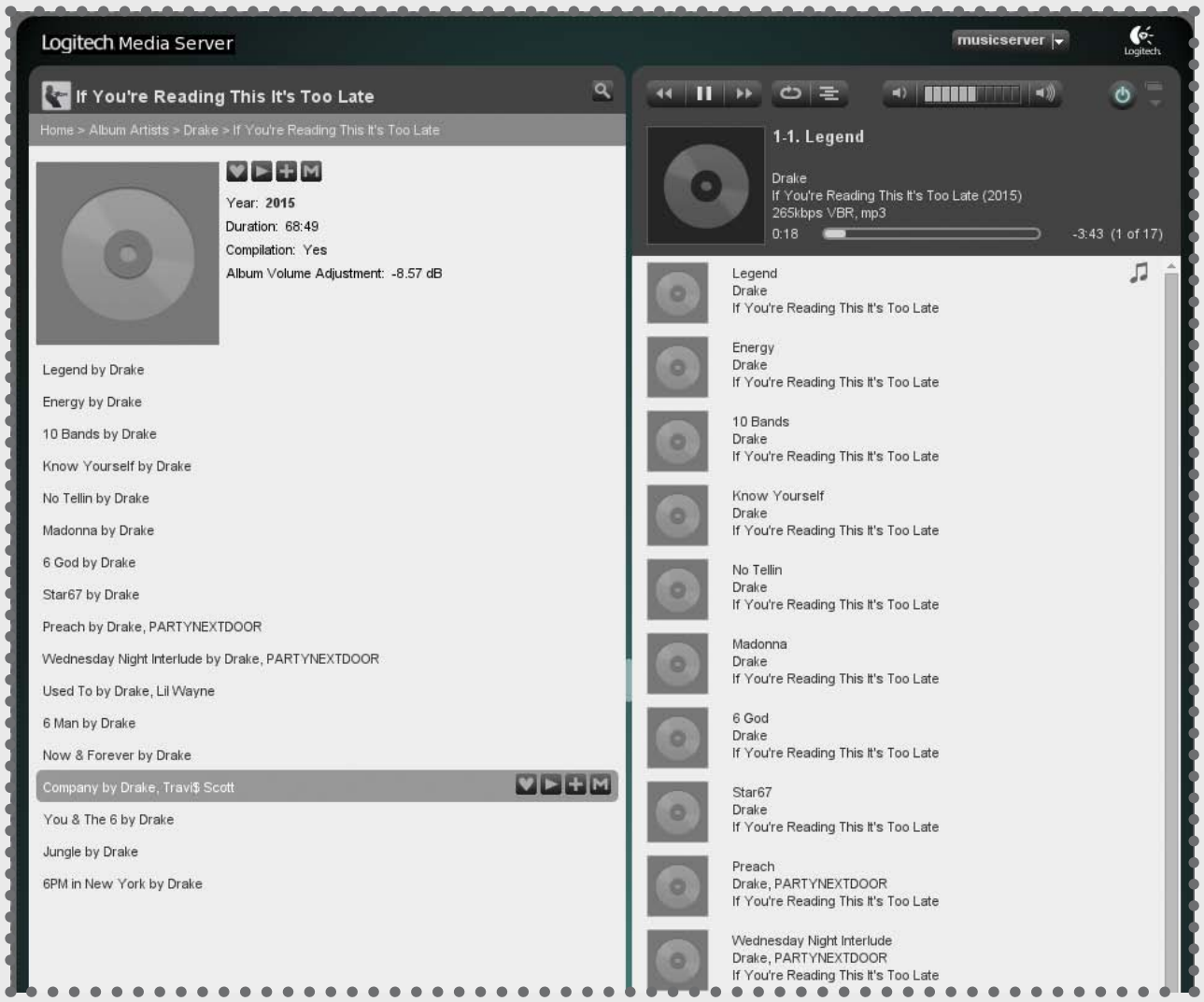
10 Configure Logitech Media Server

Now we have copied music to the server, it's time to configure Logitech Media Server. Logitech Media Server runs on port 9000, we had to go to **http://musicserver:9000/** in a web browser to access the configuration wizard. Depending on your network setup you may need to use the IP address of the server. You'll need to create a mysqueezebox.com account before you can continue. Once you have done that, log into the server. Then navigate to where your music collection is stored. This should be /mnt/music. Click next, also set /mnt/music as the playlist folder even though there probably aren't any playlists in there. Finally, click Finish. If you ever need to rescan your library you can do that by going to settings (in the bottom right corner) and then click the rescan button.

11 Play music

Playing music from the web browser is straightforward. On the left hand side is your music library, and on the right-hand side is the current playlist. In the top right corner you can pick between players;

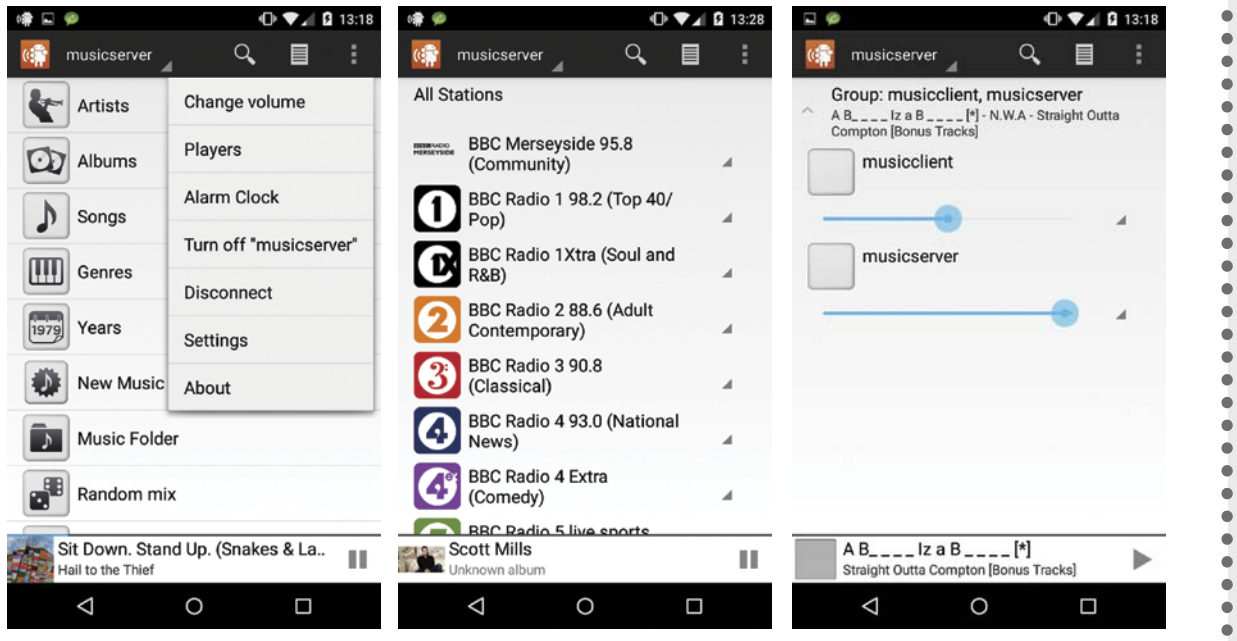
an RCA cable to connect the sound card to the AMP. The cost of the bookshelf speakers depends on the quality that you are after. Alternatively, you could even get a set of 2.0 or 2.1 PC speakers with the amp built in. You can get these for £25 on the cheap end, which means you just need a Pi with network connectivity, and a sound card.



we have musicserver, and musicclient, but you could name these as the rooms in your house. There is also a synchronise button which lets you group various rooms together. Then whatever you play will play in multiple rooms. You can also control the volume of each room using the volume control. This means you can leave your amp turned up fairly loud and then control the volume from your phone or browser.

12 Control your music from a smartphone

For Android, we recommend the Squeezer app, which can be found by searching for "squeezebox" on the Play store. Similarly, there is also an app for Apple devices. The app should automatically detect the server and connect to it. From there, the drop-down box in the top-left corner will let you select the player and the



volume rocker on the phone will control the volume of the playback.

13 Listen to the radio

Logitech Media Server is capable of playing internet radio. There are several stations to pick from, sorted by genre, area, and so on. These can be played from both the mobile application and the web interface.

14 Group players on smartphone

To group players on your phone, press the menu button in the top right corner and then select players. Then pick the player you want to group, click the synchronise button and pick the grouping you would like from the list. To desynchronise, do the same thing but pick "No grouping". You can also individually set the volume of each room from the players menu.

15 Configure podcasts

If you want to listen to podcasts then you need to enable the "Podcasts" plugin in the settings page of Logitech Media Server. Tick the plugin and click the

Apply button. You will then need to restart Logitech Media Server, which you can do with:

```
sudo systemctl restart logitechmediaserver
```

The web interface may take a few seconds to come back so don't worry if this is the case.

16 Add podcast RSS feed

To add a podcast to listen to you will have to find the RSS feed of the podcast. Our expert is a fan of The Smoking Tire, which is a podcast about cars and driving. To add a podcast feed, you need to go to the Settings page of Logitech Media Server, then to the Podcasts tab. Once there, paste in the URL for the RSS feed of your podcast.

17 Test podcast playback

Podcasts show up in My Apps > Podcasts, and work from both the web interface and the mobile app. As you can see here, a list of podcast episodes sorted by most recent first will be shown, you simply have to click on them and press play.

Pixel Vision

Love Hultén's retro-styled handheld gaming system brings CRT technology into the 21st century

What inspires you to create the work you do? Tell us about your process.

In my work I fuse modern technology with traditional crafting, creating a balanced blend of function and aesthetics. Over the past couple of years I've dedicated myself to making unique gaming consoles that cover different shapes and sizes.

These consoles are often personal tributes to the golden age of arcades, where great games were created out of limitations and big physical buttons gave us instant feedback and total control; a sense of interaction that differs a lot from today's flat shiny surfaces and touchscreens.

In my designs, I play a lot with historic norms and standards, and I give objects new functions, new values. The smashed-up references in my work have a triggering effect on the customer, I guess. I want my audience to be enlightened – not just feel nostalgic. Nostalgia is involved to a certain extent, yes, but it's not looking backwards. It's taking steps in different directions simultaneously by using fragments from both the past and today, creating unique and balanced objects.

How long have you been working on Pixel Vision for – how long has it taken to get it into production?

Pixel Vision started as a small personal project titled PE358. I travel in my work, and wanted a gaming portable for the road. The PE358 prototype finished in February 2015, and when I posted some pictures on the web, people were clamouring for me to mass-produce them. The feedback there was very generous, and I decided to give in. I started re-designing PE358 in July, and had a final design, under a new name, in October last year.

I have always wanted to try out Kickstarter, and this project seemed like a good candidate. So I launched, and

Love Hultén's passion for retro gaming, stunning design and the Raspberry Pi has culminated in some of the most breathtaking handheld consoles in recent times.

the campaign ended successfully with about 150 Pixel Vision orders. I'm a perfectionist and I need to be in total control. That means dealing with all progress stages myself personally, all the way from concept design to shipping logistics. The custom Pixel Vision PCB board is designed by me and my programmer friend, and is made locally in Sweden. The walnut enclosures are being made by me in my workshop. Through the Kickstarter campaign, there were also a limited number of units that had a mother-of-pearl veneer.

How have you found working with the Raspberry Pi for this project?

I've been working with the Raspberry Pi since 2013, and feel very comfortable using it. I like the form factor, and since I needed something with small dimensions, it was the given board for this project. I decided I was going to work with the smallest board at hand, the A+ model. During my Kickstarter campaign in November, they announced the new, even smaller board; the Pi Zero. I was dying to use it for this

THE PROJECT ESSENTIALS

Raspberry Pi A+ board
DAC+ audio chip
3.5" LCD monitor
LiPo 2000 mAh battery
LED indicators
Walnut wood shell

project, but I had a deadline and the first Zero copies were all sold out. I will use them for future projects though, no doubt. The Raspberry Pi is one of the easiest boards to program, so applying an emulator onto it took a matter of seconds.

Would you say RetroArch is one of the best emulators? Does it work well with the Raspberry Pi?

RetroArch is the only one I've used for the Raspberry Pi so far. It does the job, and it does the job well. I have no complaints. There are other alternatives out there for Raspberry Pi users though, so if someone is looking to build their own console with the Pi, I implore them to do a bit of research and see what options are available to them.

Could you tell us more about how you were able to re-create that old style CRT monitor effect on the Pixel Vision?

Most curvature shaders (CRT-simulation) found out there are quite demanding, and applying these filters onto the Pi

simply does not work. Usually this is down to the hardware requirements for such filters. Most games will run really slow or not at all. But I got help from someone working in graphic programming, who managed to create a flawless custom curvature shader for me. The finished effect is brilliant and it helps bring [to life] that old CRT effect that was a big part of these classic games.

Do you think it's possible to recreate this project on a bigger screen?

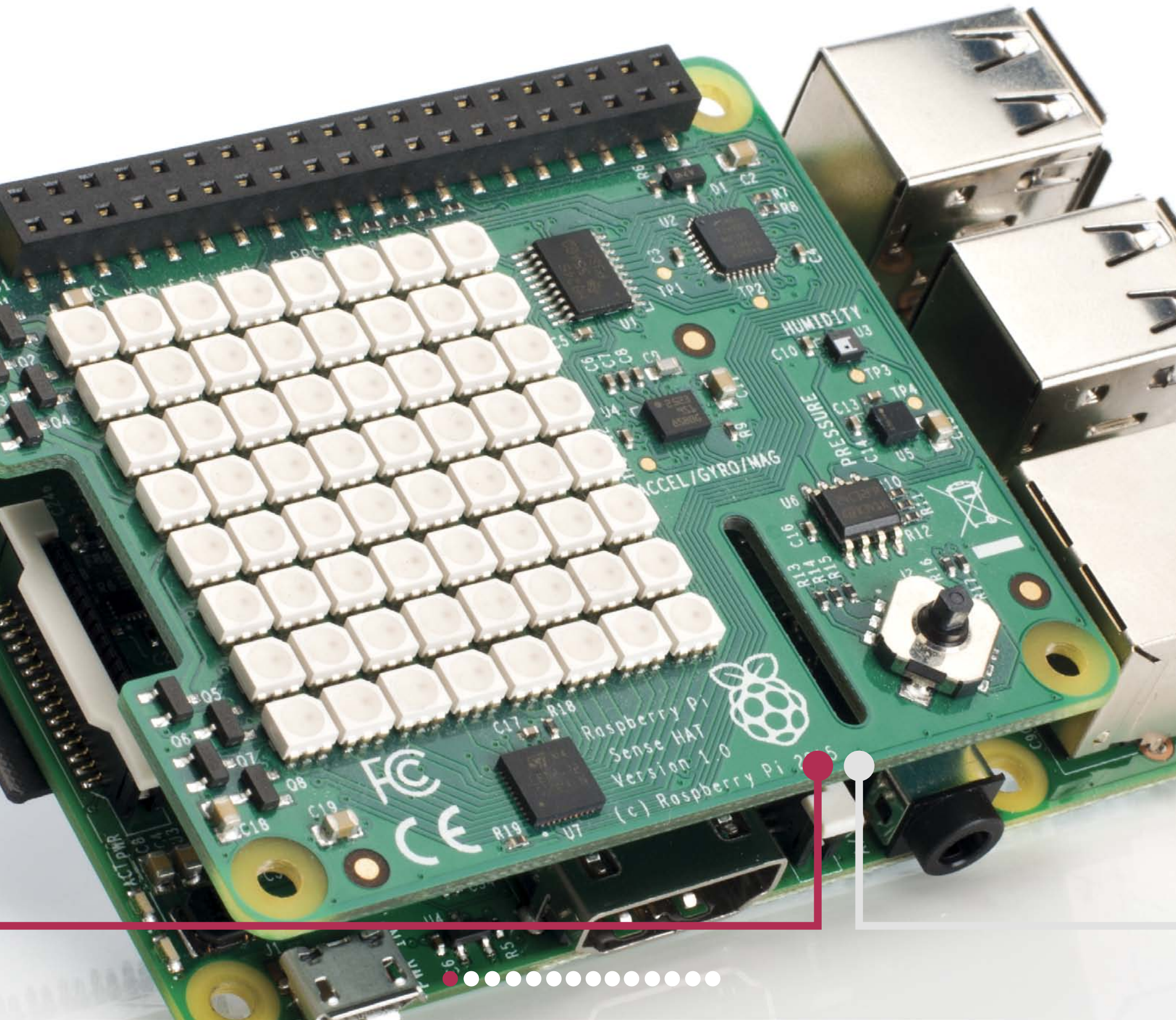
My two bigger consoles, R-Kaid-R and Pixelkabinett 42, they use the very same curvature shader. That's a yes, I guess.

So what's next for you? Do you have other exciting Raspberry Pi projects in the works?

I always have a new Raspberry Pi project up my sleeve, and look forward to working with the Zero board. The form factor is amazing. But I cannot tell you anything about future projects, that would spoil all the fun!

Master essential Sense HAT skills: part 2

Use your skills from last issue's tutorial to create a Sense HAT version of RPSLS



You will probably have played *Rock, Paper, Scissors* in real life. The issue with this version is that there are only three possible outcomes other than a tie. Sam Kass and Karen Bryla invented an alternative version which adds "Spock" and "lizard". "Spock" is signified with the Star Trek Vulcan hand sign, while "lizard" is shown by forming the hand into a mouth. Spock smashes scissors and vapourises rock; he is poisoned by lizard and is disproved by paper. Lizard poisons Spock and eats paper; it is crushed by rock and decapitated by scissors. This tutorial walks you through creating your own SenseHAT version of the game.

THE PROJECT ESSENTIALS

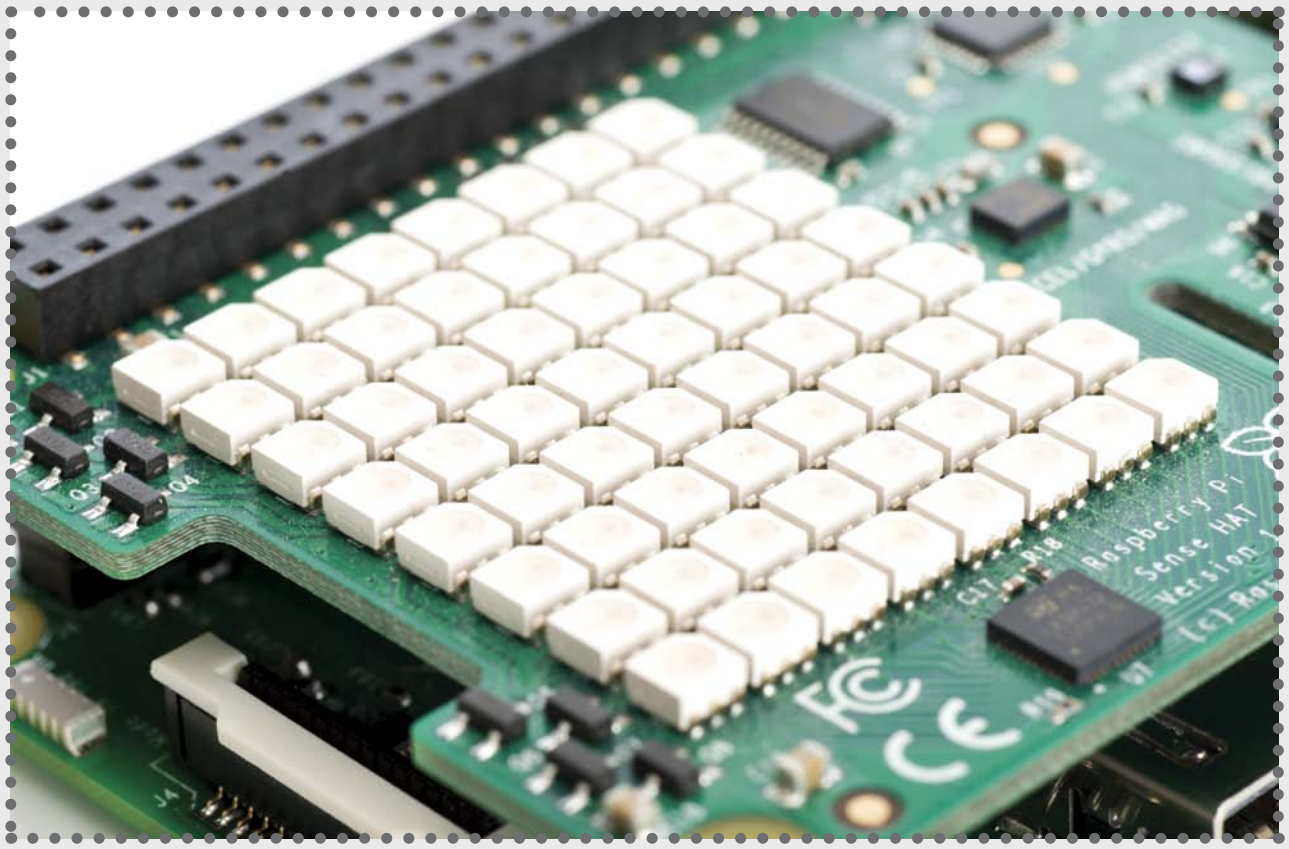
Wireshark
(www.wireshark.org)
Raspberry Pi 2
SenseHAT

01 How RPSLS works with the Sense HAT

When you complete this tutorial you will have a folder which contains five 8 x 8 images, each representing one of the items: rock, paper, scissors, lizard or Spock. These images are named 0.png, 1.png, 2.png, 3.png and 4.png. When the program runs, it will welcome you and ask you to use the joystick to select an item. Each time you push the joystick up, the item will change. Press enter to select. The computer then selects a random item. Each picture is assigned a value and a formula is used to find the modulus (the remainder) of dividing the two values. The value of the remainder determines the outcome: win, lose or draw.

02 Import the modules

Boot up your Raspberry Pi and open the LX terminal. Type sudo idle3 to load the Python 3 editor. Import the pygame module (line one) and the SenseHAT module (line three). The program uses the random module to select the computer's choices in the game. Import the



random module (line four).

```
import pygame
import pygame.locals import *
from sense_hat import SenseHat
import random
import time
```

03 Scrolling a message

Throughout the game the player is updated via messages which are scrolled across LED matrix. The SenseHAT API simplifies the whole procedure to a simple line of code: sense.show_message("This is a test message"). Your message will be scrolled across the SenseHAT LEDs. Change the text between the quotation marks and add your own message. Adjust the colour of the message and the time it takes to scroll by including the lines, text_colour=[255, 0, 0]) (setting the RGB value) and scroll_speed=(0.05), Try experimenting with the example code:

```
sense = SenseHat()
sense.show_message("Linux User &
Developer", text_colour=[255, 0, 0])
```

04 Mapping an LED image

Images are made of pixels that combine to create an overall picture. Each LED on the matrix can be automatically set from an image file. For example, an image of a lizard can be loaded, the colours and positions calculated and then the corresponding LEDs enabled. The image needs to be 8 x 8 pixels so that it fits the LED matrix. Download the test picture file – lizard.png – and save it into the same folder as your program. Use the code below to open and load the image of the lizard.

```
from sense_hat import SenseHat
sense = SenseHat()
sense.load_image("lizard.png")
```

05 Create the game variables and initialise PyGame

Next, create variables to store the player's choice, line one, the computer's choice, and also to track the picture number that is currently displayed on the LED matrix, global count. These are set as global variables that enable them to be accessed within other parts of the program and to return the values that are stored. Now, set up the PyGame window typing pygame.init() and pygame.display.set_mode((140, 180)). The window is not used in the game so set it to a small size. Load the first image with the code sense.load_image("0.png"), this loads picture zero from

your folder. You may find that the LEDs are too bright, add sense.low_light = True, to reduce the brightness. Finally set playersChoice to 0, the first image, and then set the game running with gameRunning = True.

```
global playersChoice
global computer_choice
global count
###Set up PyGame Screen###
pygame.init()
pygame.display.set_mode((140, 180))

###Prepare Sense Hat###
sense = SenseHat()
sense.load_image("0.png")
sense.low_light = True #save your eyes!

playersChoice = 0
gameRunning = True
```

06 Convert the values / numbers into items

During the RPSLS gameplay the program uses numbers to identify the items instead of their names. This means you can select a picture and also use a modulus operation to calculate the outcome of the game. Create a function that converts and assigns the value into the respective item name. A simple conditional checks what the number is and returns the name of the item. The value of 'scissors' is set twice – during the gameplay a loop checks that the fifth image has been loaded and then resets the

Pixel inspiration

You will need an image for Rock, Paper, Scissors and Spock. Johan Viet has some excellent and inspirational examples of 8 x 8 pixel art. Check them out at **http://johanvinet.tumblr.com/image/127476776680**

value to zero which would result in the fifth image not being displayed.

```
def number_to_name(number):
   if number == 0:
      return “Rock”
   elif number == 1:
      return “Spock”
   elif number == 2:
      return “Paper”
   elif number == 3:
      return “Lizard”
   elif number == -1: ### because value is
5 so re-sets to 0, zero - 1 = -1 ###
      return “Scissors”
   elif number == 4: ### because value is
5 so re-sets to 0, zero - 1 = -1 ###
      return “Scissors” ### for the
computer
```

07 Check for joystick movement

Create another function that holds the main mechanics of the gameplay. Start by adding the global variables sense.set_rotation(90), which positions the image correctly. Next use for event in pygame.event.get(): to check for a joystick movement. The joystick is used to cycle through a loop of the five LED images. The line if event.type == KEYDOWN and playersChoice < 5: checks that the player has moved the joystick up and also that the picture number is less than 5. If it is equal to five then the loop resets. If the playersChoice value is less than five then the

corresponding picture number is loaded to the LEDs using sense.load_image(str(playersChoice) + ".png") line 13 and the variable incremented. The loop restarts from the beginning and checks for a joystick movement and then displays the relevant picture file.

```
def mainGame():
   ###PLAYER SELECTION###
   ###Loops while running variable is
True###
   running = True
   global playersChoice
   global computer_choice
   while running == True:
      sense.set_rotation(90)
      for event in pygame.event.get():

         if event.type == KEYDOWN and
playersChoice < 5:
            if event.key == K_UP:
             print (playersChoice)
              sense.load_
image(str(playersChoice) + ".png")
              playersChoice = playersChoice +
1
         if playersChoice == 5:
           playersChoice = 0
```

08 Select your item

When you are scrolling through the images, you need to be able to select an item to play. Press the joystick down to act as the Return key. Add the line if event.key

== K_RETURN: line three, to respond to the joystick being pressed down, then end the picture cycle loop using running = False. Finally 'break' out of the loop to move onto the next part of the game.

```
'''Checks for a 'select / Enter' Choice '''
        if event.type == KEYDOWN:
            if event.key == K_RETURN:
                running = False
                break
```

09 Update the status

Once you have made your selection, scroll a confirmation message across the LEDs. Create a new variable called number and assign the playersChoice value to it. Next, use the function in step six to convert the number into the name of the item, playerMessage = number_to_name(number). Finally scroll the message across using the line, sense.show_message(playerMessage, text_colour=[0, 0, 255], scroll_speed = 0.08). You can change the text colour by altering the values within the square brackets and also the speed of the scroll between the values of zero and one.

```
'''Message for player about their choice'''
    number = playersChoice - 1
    playerMessage = number_to_name(number)
    print playerMessage
    sense.set_rotation(0)
    sense.show_message("You = ", text_
colour=[0, 255, 255], scroll_speed = 0.08)
```

```
    sense.show_message(playerMessage, text_
colour=[0, 0, 255], scroll_speed = 0.08)
```

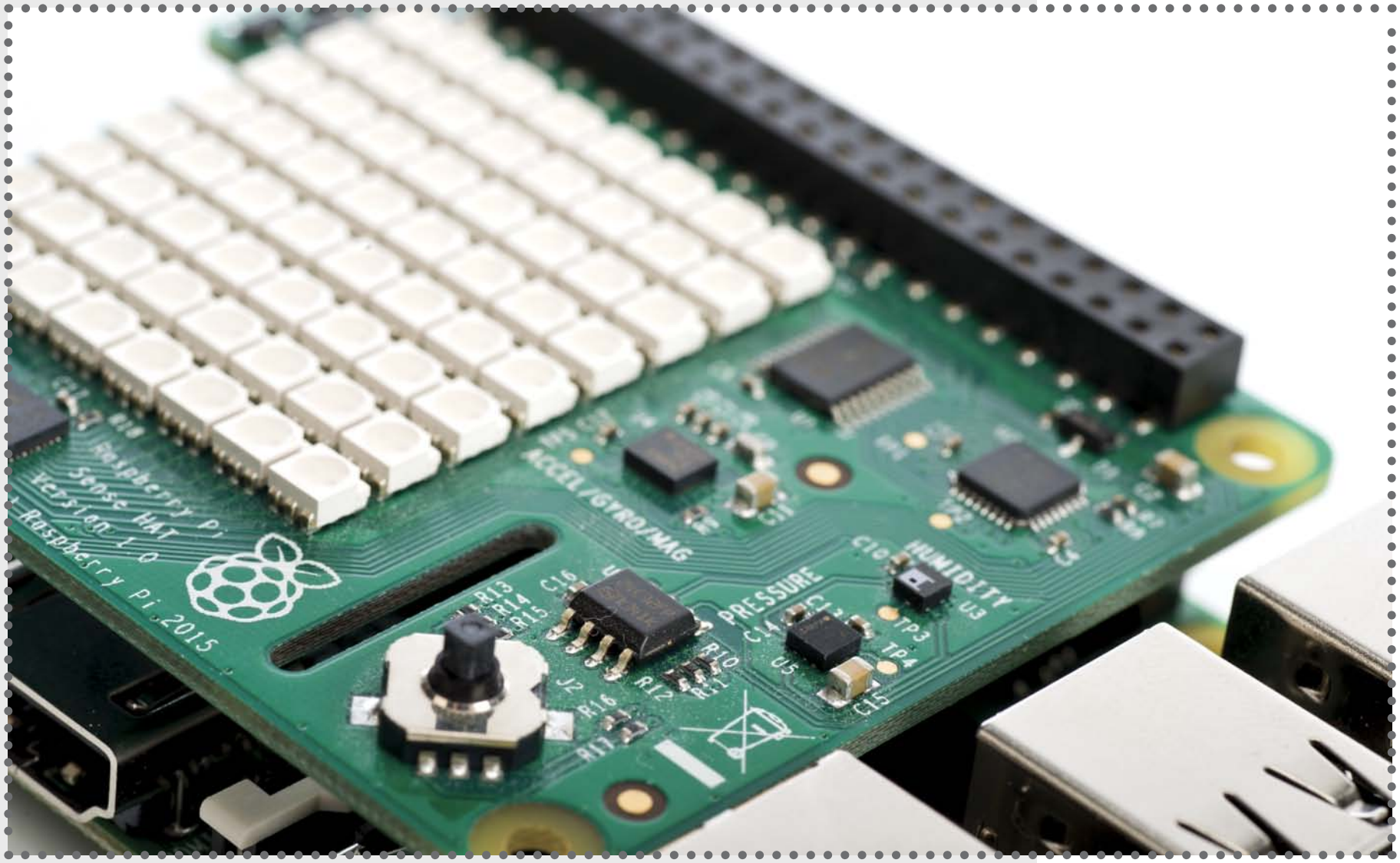
10 The computer's selection

Now it's the computer's turn. First the computer selects a random number between 5 and 50, count = random.randrange(5,50). This is a random number of times that it will cycle the choices to give the impression of 'thinking'. Then the computer selects a random number between 0 and 5, computer_choice = random.randrange(0,5). This is the computer's 'item' selection. Then one is subtracted from the cycle and the computer selects a new picture. This continues while the value of count is greater than zero. In each cycle the picture is displayed on the LED matrix with sense.load_image(str(computer_choice) + ".png")

```
###COMPUTER SELECTION
    '''Computer selects a random choice from
the options'''
    count = random.randrange(5,50)
    sense.set_rotation(90)
    while count > 1:
        computer_choice = random.
randrange(0,5)
        print computer_choice
        time.sleep(0.1)
        sense.load_image(str(computer_choice) +
".png")
        count = count - 1
```

11 Game status update

The next part is to use messages to update the



player. Use the function in step six to look up and convert the number into an item: computerMessage = number_to_name(number). Now use sense.show_message("Computer = ", text_colour=[0, 150, 255], scroll_speed = 0.06) to display the computer's final choice:

```
#time.sleep(1)
    '''Message for player about the
computer's choice'''
    print ("The computers choice is",
computer_choice)
    number = computer_choice
    computerMessage = number_to_name(number)
    print computerMessage ##test
    sense.set_rotation(0)
```

```
    sense.show_message("Computer = ", text_
colour=[0, 150, 255], scroll_speed = 0.06)
    #sense.show_message(computerMessage,
text_colour=[0, 0, 255], scroll_speed =
0.08)
    sense.load_image(str(computer_choice) +
".png")
```

12 Did you win?

The value of the computer's choice is subtracted from the value of the player's choice and divided by five, returning the remainder to determine the outcome of the game. Create a new variable called result and assign the calculation to it, result = (int(computer_choice - (playersChoice-1))) % 5. If the result is zero, then the player and the computer have picked the same item: the game is a tie. Use an IF statement on line five to check the value and then display a message using sense.show_message("Player and Computer Tie!", text_colour=[0, 0, 255], scroll_speed = 0.08).

```
###WINNER CALCULATED###
    '''Calculates the Winner'''
    result = (int(computer_choice -
(playersChoice-1))) % 5
      #print result
    if result == 0:
        sense.show_message("Player and
Computer Tie!", text_colour=[0, 0, 255],
scroll_speed = 0.08)
```

To win, the value of the remainder has to be greater than or equal to three. Use an ELSE IF statement to check this, elif result >=3. Any other value means that the computer has won this round. Add an else statement and use sense.show_message("Computer Wins!", text_colour=[255, 0, 0], scroll_speed = 0.08 to display the winning or losing messages.

```
elif result >=3:
        sense.show_message("Player Wins!",
text_colour=[0, 255, 0], scroll_speed =
0.08)
 else:
        time.sleep(1)
        sense.show_message("Computer Wins!",
text_colour=[255, 0, 0], scroll_speed =
0.08)##??
```

13 Introducing the game

Now to add an intro message and instructions. You could add these into the main game loop but each time you start a new round you have to wait for them to repeat. Type sense.show_message("Welcome to RPSLS!", text_colour=[155, 100, 30], scroll_speed = 0.08) to scroll the messages and load the first image onto the LEDs with sense.load_image("0.png").

```
###START THE GAME##
sense.show_message("Welcome to RPSLS!",
text_colour=[155, 100, 30], scroll_speed =
0.08)
sense.show_message("Pleases use 'Up' to
```

Creating the 8 x 8 images

A simple method to create your images is to use the RPi Grid Draw program. This enables you to manipulate the LEDs in real time. You can change the colours, rotate them and then export the image as code or as an 8 x 8 png file. Install 'Python PNG library', open the Terminal window and type: sudo pip3 install pypng. After this has finished type, git clone **https://github.com/jrobinson-uk/RPi_8x8GridDraw**. Once the installation has completed move to the RPi folder, type cd RPi_8x8GridDraw, type python3 sense_grid.py to run the application

```
select", text_colour=[155, 255, 255],
scroll_speed = 0.05)
sense.load_image("0.png")
```

14 Detecting play

Create a while loop that checks that the game is running. If it is, the game is in play. Next set a variable called play_again to a value of one. This is used later on to end the game or to play another round. Then call the function created in steps seven which holds the game mechanics, on line three, type mainGame(). When the round finishes scroll a message across the LEDs asking the player if they want to play again, sense.show_message("Play Again?", text_colour=[255, 255, 255], scroll_speed = 0.08) line four.

```
while gameRunning == True:
    play_again = 1

    mainGame()
    sense.show_message("Play Again?", text_
colour=[255, 255, 255], scroll_speed =
0.08)
```

15 Play again?

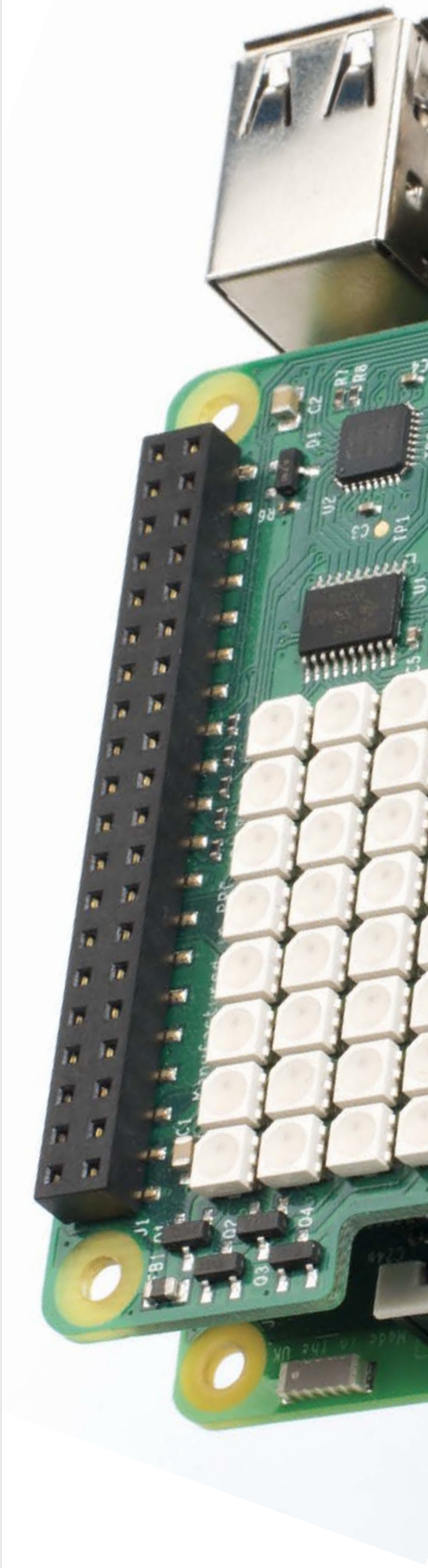
To respond to the 'Play Again' question, use an IF statement to select the option to play again. This is enabled by moving the joystick up. Use for event in pygame.event.get(): on line two and check for a 'keydown' event, the joystick being moved up, if event.type == KEYDOWN:, if event.key == K_UP: lines three and four. If the Up is selected then the play again

variable is set to zero, line five, which ends the IF statement, and starts the game again.

```
while play_again == 1:
    for event in pygame.event.get():
        if event.type == KEYDOWN:
            if event.key == K_UP:
                play_again = 0
```

16 End the game and exit

Code this option to quit by checking for a joystick movement down, if event.key == K_DOWN: Type a message using sense.show_message("Bye Bye", text_colour=[255, 255, 255], scroll_speed = 0.08) and change the gameRunning variable to False. When the program loops back to the start it will find that the loop from step 14 is now False, so the game will stop running, as you can see in the last section of code. Enjoy playing!



Networked Sensor Display with Pimoroni Scroll pHAT

Pair up two Raspberry Pis to create a live networked sensor display

We are going to create a networked sensor display with the Pimoroni Scroll pHAT. We'll take a look at the hardware, what's included and how to install the required Python libraries. Then we will tour the library's source code and code some examples that have been contributed by the open-source community through GitHub. Once we are familiar with the library, we will go on to set up our chosen sensor on the first Raspberry Pi and connect the screen to our Pi Zero. We'll learn about how to use Redis (which is an open-source pub-sub noSQL database).

Redis is our connective tissue, publishing events that our Zero will subscribe to and then display on the Scroll pHAT. This saves on the complexity of implementing a web-server or going even lower level with sockets. Don't worry if you do not have a Scroll pHAT – we've thought of that and provided a version of the code that displays text on your SSH terminal instead.

THE PROJECT ESSENTIALS

Article's Github repository (http://github.com/alexellis/rpi-display)

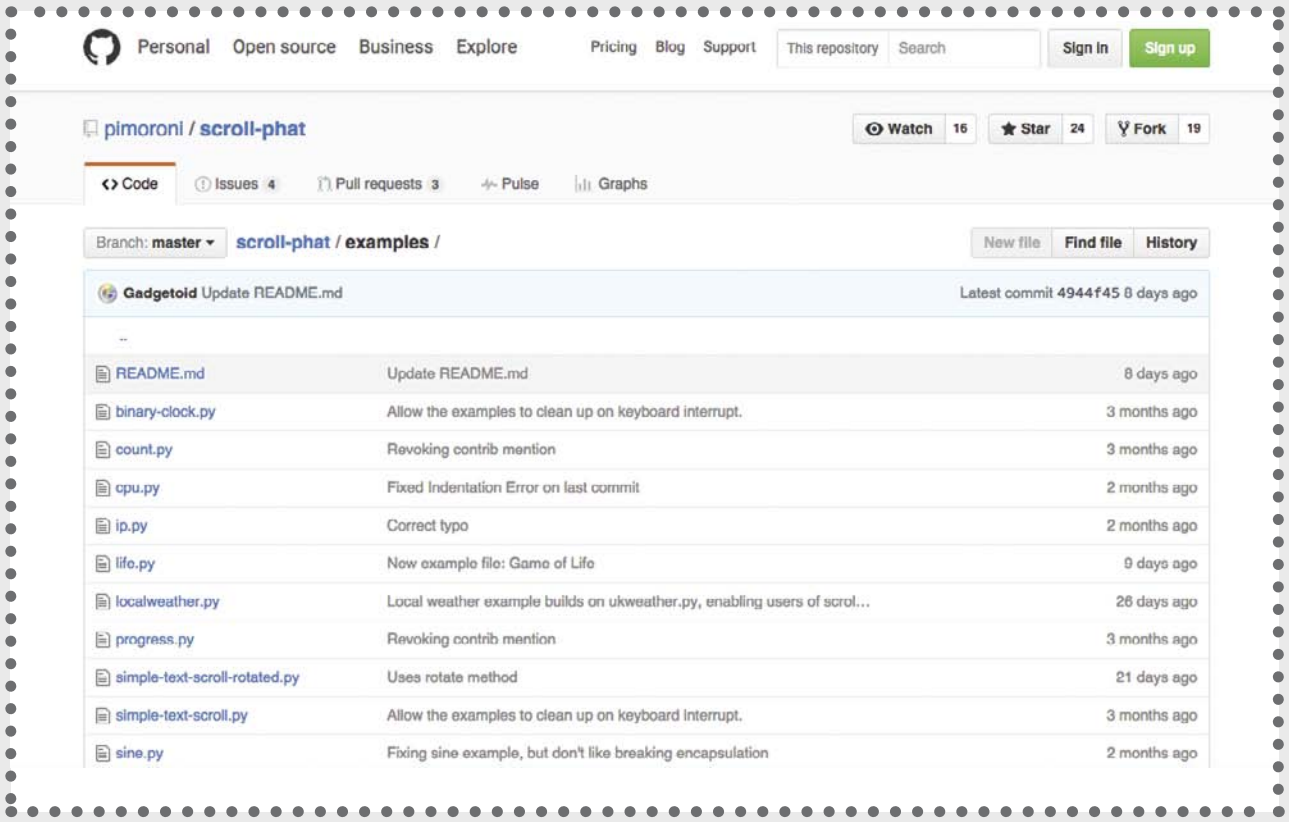
Two Raspberry Pis

Pimoroni Scroll pHAT (optional)

Soldering iron and solder

01 First look at the Scroll pHAT

The Scroll pHAT is one of a number of pHATs (hardware attached on top) which appeared at around the launch of the Raspberry Pi Zero. Pimoroni's device costs £10 when ordered directly from the website and is also available in kit form, which is excellent value considering that the price of an empty PCB can be around £5. For your money you get an eye-catching PCB in a black finish with 11 rows of 5 LEDs pre-mounted. At the heart of the pHAT is ISSI's IS31FL3730 LED Matrix driver. The LED driver speaks the I2C protocol, making it capable of connecting to almost any micro-controller such as the Arduino.



02 Solder the 40-pin header

Pimoroni's device is almost ready to go, but they have left some soldering down to us. We get to solder the header pins, choosing whether to go with a vertical header or a 90-degree version (provided). Another option is to use a single set of pins and solder the Scroll pHAT directly on top of the Pi Zero making for a cute, low-profile sandwich of tech. Whichever option you go with, make sure that you have a well-ventilated environment with good lighting. We find that a flux pen and thin solder helps, too.

03 Connect the hardware

Carefully line up the Zero and the Scroll pHAT and gentle push them together. Then turn on the power and boot into Raspbian. From here open a terminal window or ssh connection and type in the following to invoke the automatic installer:

```
curl -sSL get.pimoroni.com/scrollphat | bash
```

Pro tip: If running a bash script directly over the internet scares you, then you are not alone: you can always open the script in a web browser to read through the setup process.

04 Run your first example script

Now that you have the library installed, clone the git repository, and look in the examples folder.

```
git clone https://github.com/pimoroni/scroll-phat
cd scoll-phat/examples/
```

The author has contributed several code examples, try out the progress.py example with: sudo python progress.py. You should see an animation reminiscent of the classic snake game. There are a dozen examples to try, like scrolling system uptime, a count-down, the UK weather and a binary clock.

05 Light up pixels with the library

After importing the scrollphat module we can create animations with a combination of set_pixel(), update() and time.sleep(). Try the following to blink the first LED.

```
import scrollphat
scrollphat.set_brightness(2)
while(True):
```

```
scrollhat.set_pixel(0, 0, True)
scrollphat.update()
time.sleep(0.5)

scrollhat.set_pixel(0, 0, False)
scrollphat.update()
time.sleep(0.5)
```

Use scrollphat.set_brightness() to pick an appropriate brightness for the ambient lighting.

06 Display a short message

To output a static message use write_string(text) followed by update(). This will load a basic font where each letter takes up around 5x3 pixels, allowing for around three letters.

```
scrollphat.write_string(“FOO”)
scrollphat.update()
```

The count.py example uses write_string to count up to a given number, like a stopwatch. So, what if your string doesn’t fit into three characters?

07 Display longer messages

For longer messages, we can call the scroll() method. Each call to scroll() will move the text to the left by one pixel. So to move the letter ‘f’ off the screen call scroll() three times – or in a while loop such as:

```
scrollphat.write_string(“foo”)
while(True):
```

```
scrollphat.scroll()
scrollphat.update()
```

08 Clear the screen

Clearing the screen can be done in two ways; either by calling the set_pixel(x, y, False) method to turn off individual pixels or by calling scrollphat.clear() and scrollphat.update() afterwards. You will see a pattern in many of the examples:

```
scrollphat.write_string("foo")
try:
    while(True):
        scrollphat.scroll()
        scrollphat.update()
except KeyboardInterrupt:
    scrollphat.clear()
    scrollphat.update()
```

For a good example of this, take a look at the uptime.py file.

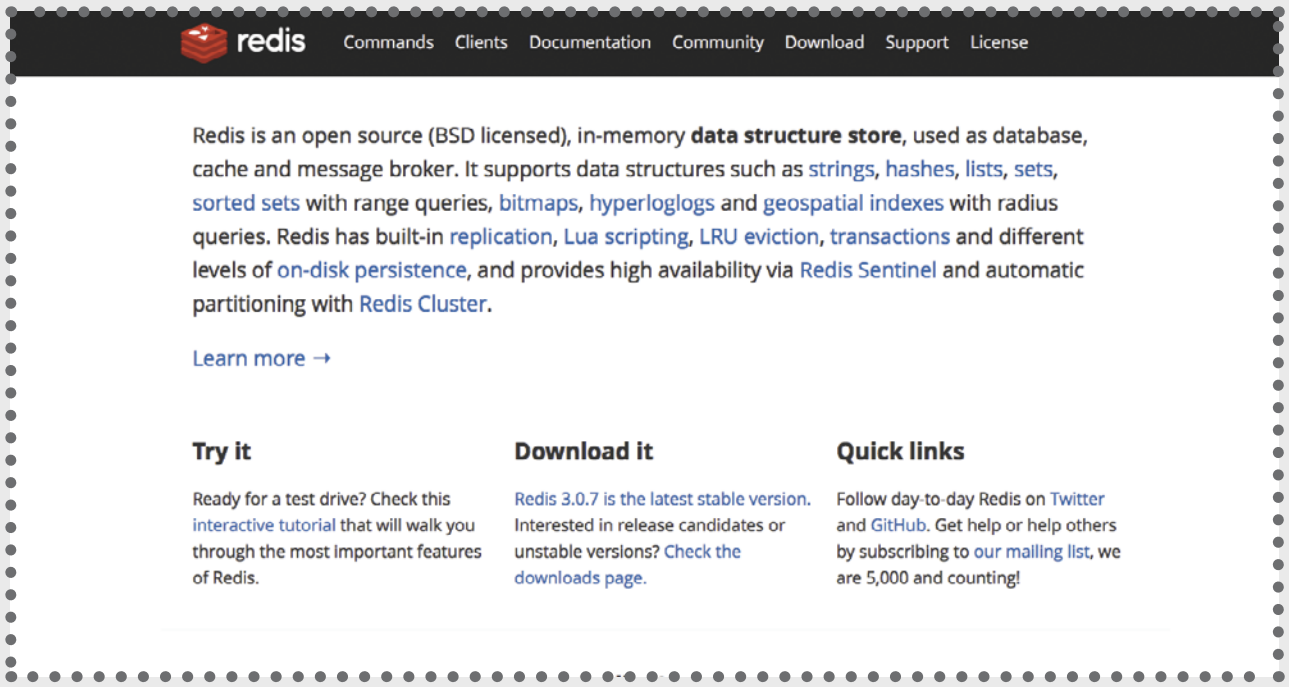
09 Connect a sensor

We will now connect a sensor to our sensing RPi, this can be any sensor with a HIGH/LOW GPIO output or something more complex as long as it has a Python library available.

The diagram generated through Fritzing shows a PIR passive infrared receiver – these can be bought for £1-3 and trigger due changes in heat signatures such as those produced by people or animals. Connect 5v to 5v, GND to GND and then

Contribute to the scrollphat library

The Scroll pHAT's Python library is open-source and available under the GPL on Github which means you can fork the repository. Forking means you can make changes in your own account and then raise a pull-request (PR) which alerts Pimoroni's developer Phil to come and have a look at your work and merge it into the main project.



the signal to pin 17 for example.

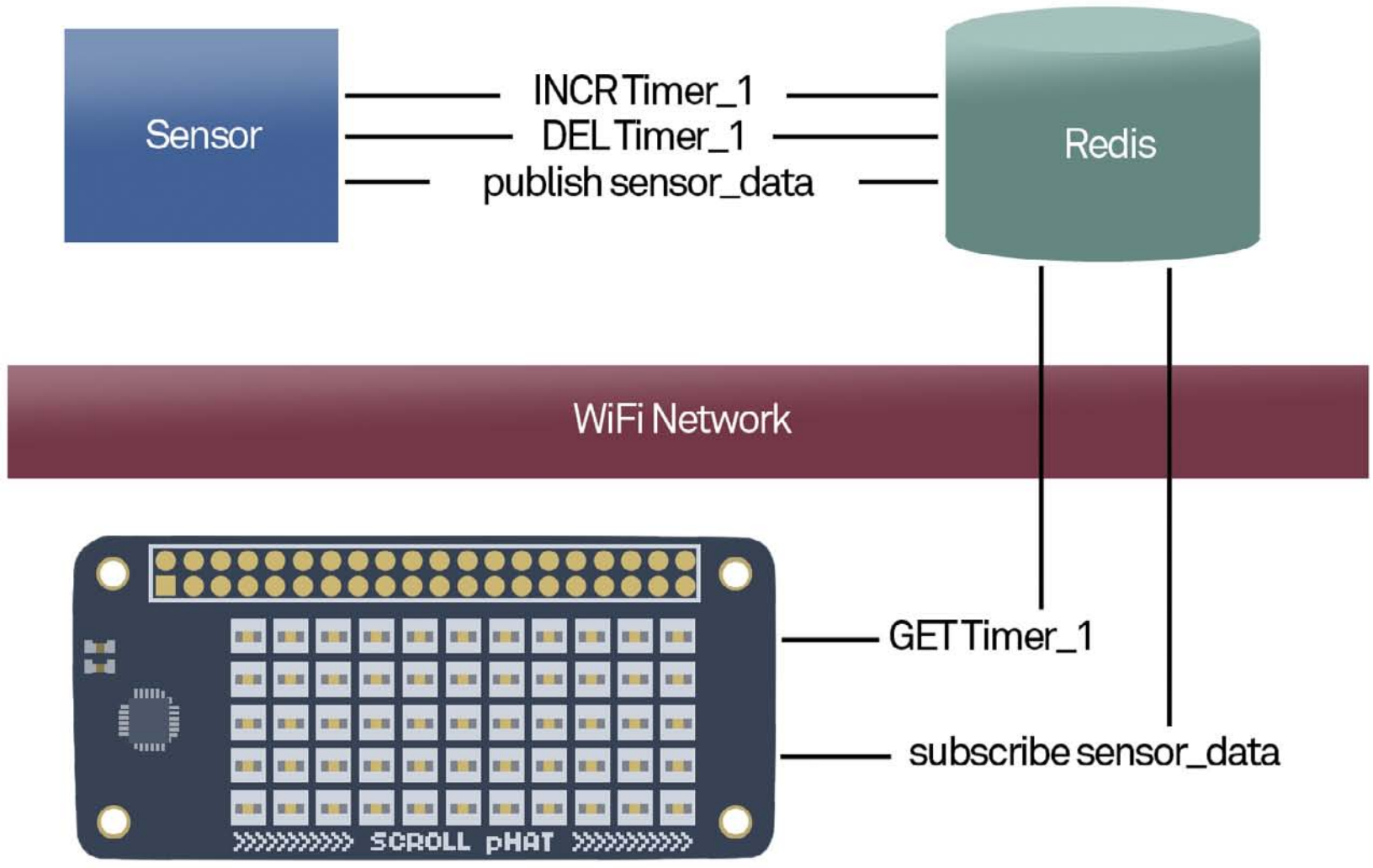
10 Install Python libraries

On both Pis, install the Redis Python library, followed by Redis itself. Redis is much more than a key/value pair store – it is capable of pub/sub, ranking, efficient set-based operations and is really fast on the Pi. The sensing Pi will interact with the display Pi's Redis instance over your Wi-Fi network on TCP port 6739. In our example we will make use of pub/sub and a key/value pair.

```
sudo pip install redis
sudo apt-get install redis-server
```

On the display Pi, configure the service to start on boot-up:

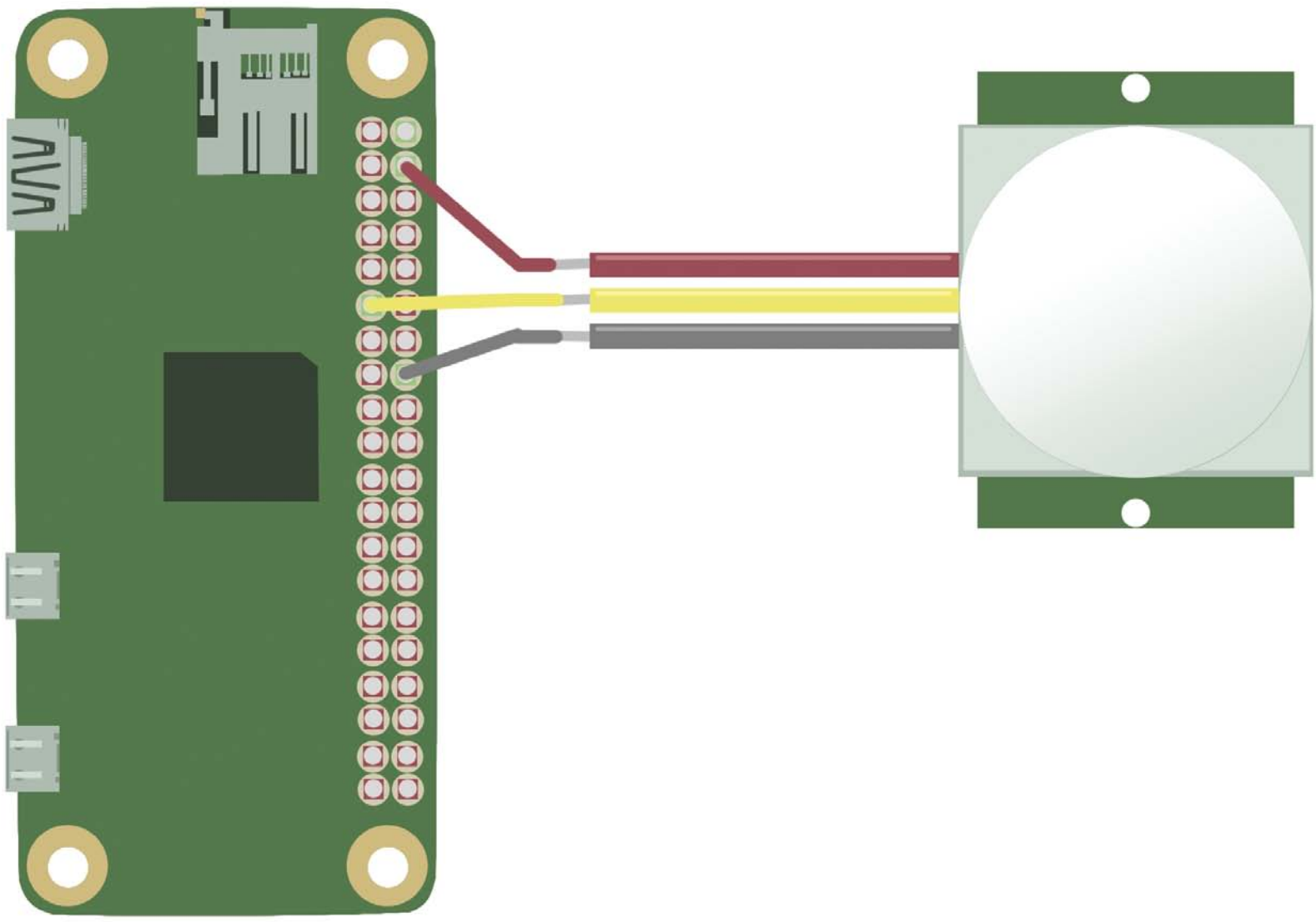
```
sudo systemctl enable redis
sudo systemctl start redis
```



11 Redis commands

The sensor.py software increments a counter for each minute of inactivity, but if the PIR senses motion it will reset the value back to zero. Whenever the counter changes, a message is published which the display unit can use to know when to refresh its output. You can also use the bundled redis-cli tool to test out these commands.

```
  * INCR <key> <amt> increments a key
by one, or the amount given
  * DEL <key> removes and zeros a key
  * PUBLISH <channel> sends a message
onto a channel
  * SUBSCRIBE <channel> subscribes to a
channel for new messages
```



12 Redis library basics

To create a client, import the Redis library and then call redis.StrictRedis, passing in the IP address of the server through the ip_address variable. Methods such as incr can then be invoked on the client for as long as you need it.

```
>>> import redis
>>> client = redis.StrictRedis(ip_
address)
>>> client.incr('counter', 10)
>>> client.incr('counter', 1)
>>> print(str(client.get('counter')))
>>> print("Counter: {}".format(client.
```

```
get('counter')))
   Counter:  11
```

We opted to create redis_controller.py; you will find this used in both sender.py and display.py.

13 Redis pub/sub

Due to the way redis works, a separate client is required for publishing and subscribing to a channel.

14 Prepare the demonstration

Start the sensor software:

- Check that the redis server has started with systemctl start redis
- Clone the github repository and CD into the folder rpi-display
- Type in sudo python sender.py

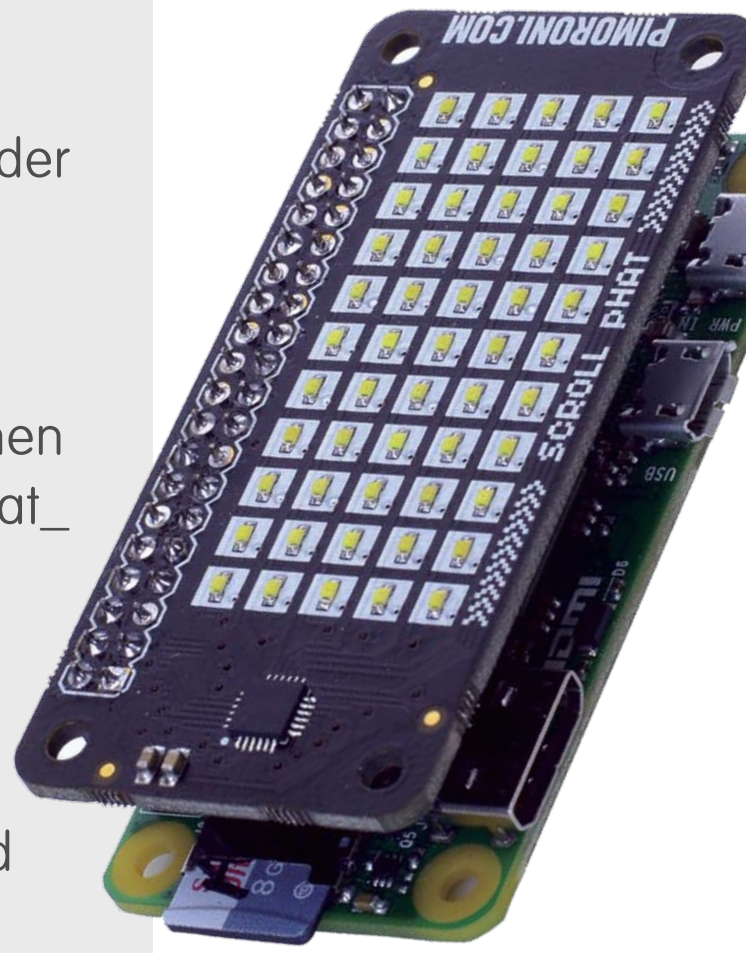
Start the display software:

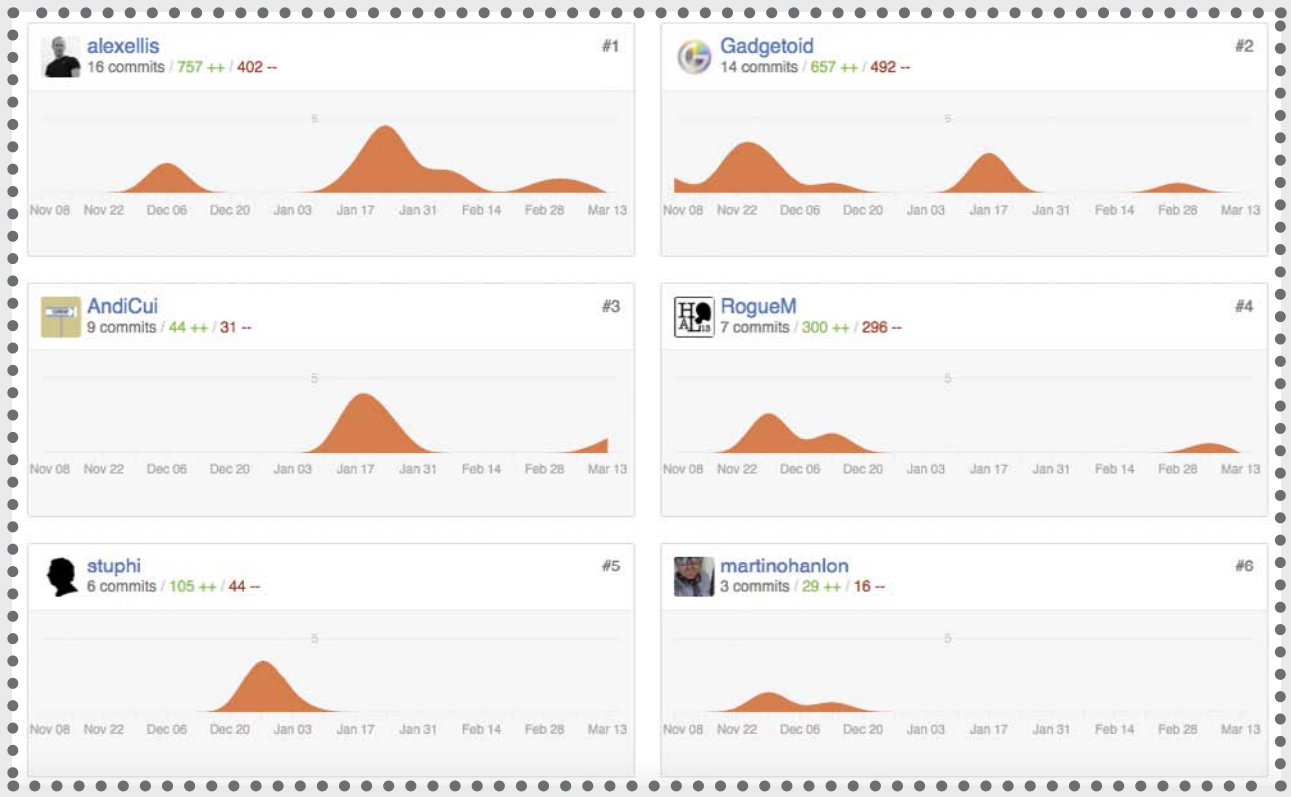
- Clone the github repository and cd into the folder rpi-display
- Type in sudo python display.py

If you are testing or do not have a Scroll pHAT then edit display.py and change "from pixel_scrollphat_display" to "from terminal_display".

15 Try out the demonstration

Now that you have started both parts of the demonstration, point the PIR away from you and





wait. As each minute passes, you should see dots accumulating on the screen of the Scroll pHAT just like a game of Tetris. When you have at least one dot on the screen, get up and walk into the pathway of the PIR – clearing the screen.

16 Show data from other sensors

The sender.py example can also be adjusted to take information from other sensors, such as the ambient temperature of the Pi's CPU. Instead of calling INCR once, you could call DEL and then INCR once for each degree before publishing the result.

If you find that the dot display does not make sense, or that your output needs to cover more than 5x11 pixels, you can use numeric_scrollphat_display.py as your display which can show a number between 0-999 without the need for scrolling.

17 Wrapping up

We have now learnt how to use the Scroll pHAT Python library to display individual pixel

animations, static text and scrolling messages. We leveraged Redis as a server and a client to provide our networking; here are some ideas for you:

- Use the PIR as a silent door bell, having it flash the screen when it detects motion
- Run a Pi web-server and display its CPU usage on the screen
- Connect a magnetic sensor to a stationary bike trainer, and show the wheel-speed through the screen

It's now up to you decide where to take it. Have fun!

Handle OpenGL on your Pi

The Raspberry Pi includes a basic GPU. Learn how to use it to generate 3D graphics on board

Many of the examples we have looked at in this column have been programs that run on the console. This issue, we will be taking a look at how you can include 3D graphics within your Python code. The hardware on the board includes a GPU, so you should be able to take advantage of everything this computer is capable of. Since we are going to be playing with graphics, this means that you will probably want to have an actual monitor connected to your Raspberry Pi. If you don't have one, you can connect to your Raspberry Pi over SSH with X11 forwarding turned on. As long as you have an X11 server on your desktop, you can see the display on your local machine. To get access to OpenGL from Python, you will need to install the correct package, with:

```
sudo apt-get install python-opengl
```

This is assuming that you are using a Debian-based distribution, such as Raspbian. It will also install the OpenGL libraries that actually talk to the GPU and handle the 3D operations. For those who don't already

know, OpenGL is the open source, multi-platform library to handle 3D graphics. For those of you coming from the Microsoft world, it is essentially Direct3D for everything non-Microsoft. All of the actual functionality is encoded within libraries written in C, while python-opengl provides a set of Python wrappers allowing you to access these libraries from your code.

The first step in trying to use OpenGL in Python is to import all of the code that you will need. Since OpenGL can be a bit messy and convoluted, there are several categories of helper functions available. These are grouped under the GLU and GLUT sub-modules. The imports that will likely be part of your boilerplate are:

```
from OpenGL.GL import *
from OpenGL.GLU import *
from OpenGL.GLUT import *
```

All of the functions that you will likely ever need should be imported to your namespace. One of the really useful functions that GLUT provides is the ability to create windows and to draw within them. In order to use it, you need to initialise the subsystem first:

```
glutInit()
glutInitDisplayMode(GLUT_RGBA | GLUT_DOUBLE | GLUT_ALPHA | GLUT_DEPTH)
```

The first thing you will need is a window to draw within. Windows need a size and a position to define where they should be drawn on the physical screen. A basic window would be defined by:

```
glutInitWindowSize(500, 400)
glutInitWindowPosition(0, 0)
```

The last step is to actually create the window. This creates a new window object and displays it on the display device in use. You can do this with:

```
window1 = glutCreateWindow('window1')
```

The newly created window is created with a title on the main window bar defined by the string handed in as an option. At this point we have a window, but there is nothing in it yet. The next step is to tell OpenGL what you want to draw on this canvas. You can do this by setting a callback function that OpenGL can use to draw with. A boilerplate of a draw function looks like:

```
def draw1():
    glClear(GL_COLOR_BUFER_BIT | GL_DEPTH_
BUFFER_BIT)
    glLoadIdentity()
    # Other drawing
    glutSwapBuffers()
```

The first line blanks out the contents of the window. The second line resets the position of the window, ready for further drawing. The commented line is where you would place any other drawing commands needed by the draw1 function. The last line is important for double-buffering. The technique of double-buffering is where you draw on an in-memory buffer region,

and then dump the entire contents of this region to the physical display device. This is done to get the fastest possible drawing speed. Once you have your function defined, you can set it as the callback function with the GLUT function:

```
glutDisplayFunc(draw1)
```

The last step is to tell OpenGL to start drawing. You can do this with:

```
glutIdleFunc(draw1)
glutMainLoop()
```

The first line tells OpenGL to run the draw1 function whenever the system is otherwise idle, In this case, this means all the time. The second line starts the processing loop for the OpenGL process. You now have a black window, which is rather boring.

Now we need to actually start drawing within this new window. In OpenGL, drawing simple objects is done by using vertices and connecting lines between them. As a simple example, we will look at drawing a basic rectangle. Any drawing commands need to be bracketed by a begin command and an end command to tell OpenGL what functions need to be paid attention to. This looks like:

```
glBegin(GL_QUADS)
# drawing commands
glEnd()
```

> "The actual drawing commands consist of telling OpenGL where the vertices are located within the window"

The actual drawing commands consist of telling OpenGL where the vertices are located within the window. For this simple case, the coordinate system of the window starts at (0, 0) in the bottom left corner. The x-coordinate increases towards the right, and the y-coordinate increase towards the top. If we have the coordinates of the bottom-left corner of the rectangle in the variables x and y, with a width and a height defined, we can set the vertices with the code:

```
glVertex2f(x, y)
glVertex2f(x + width, y)
glVertex2f(x + width, y + height)
glVertex2f(x, y + height)
```

Notice that you need to define the vertices in order, going around your rectangle. You probably also want to draw your rectangle is some other colour than the black background of the window. You can do this by changing the colour used, just before you start drawing. The command is:

```
glColor3f(0.0, 0.0, 1.0)
```

The parameters are floating point numbers, from 0.0 to 1.0, to define how much red, green and blue to use in the colour. So the above command will give you a bright blue rectangle. If you were to run this code as is, however, you still won't see your rectangle; this is because we haven't told OpenGL that we are drawing in 2D yet. A function that contains all of the required steps looks like:

Left Raspbian's OpenGL driver means we'll be able to play 3D games more advanced than this!

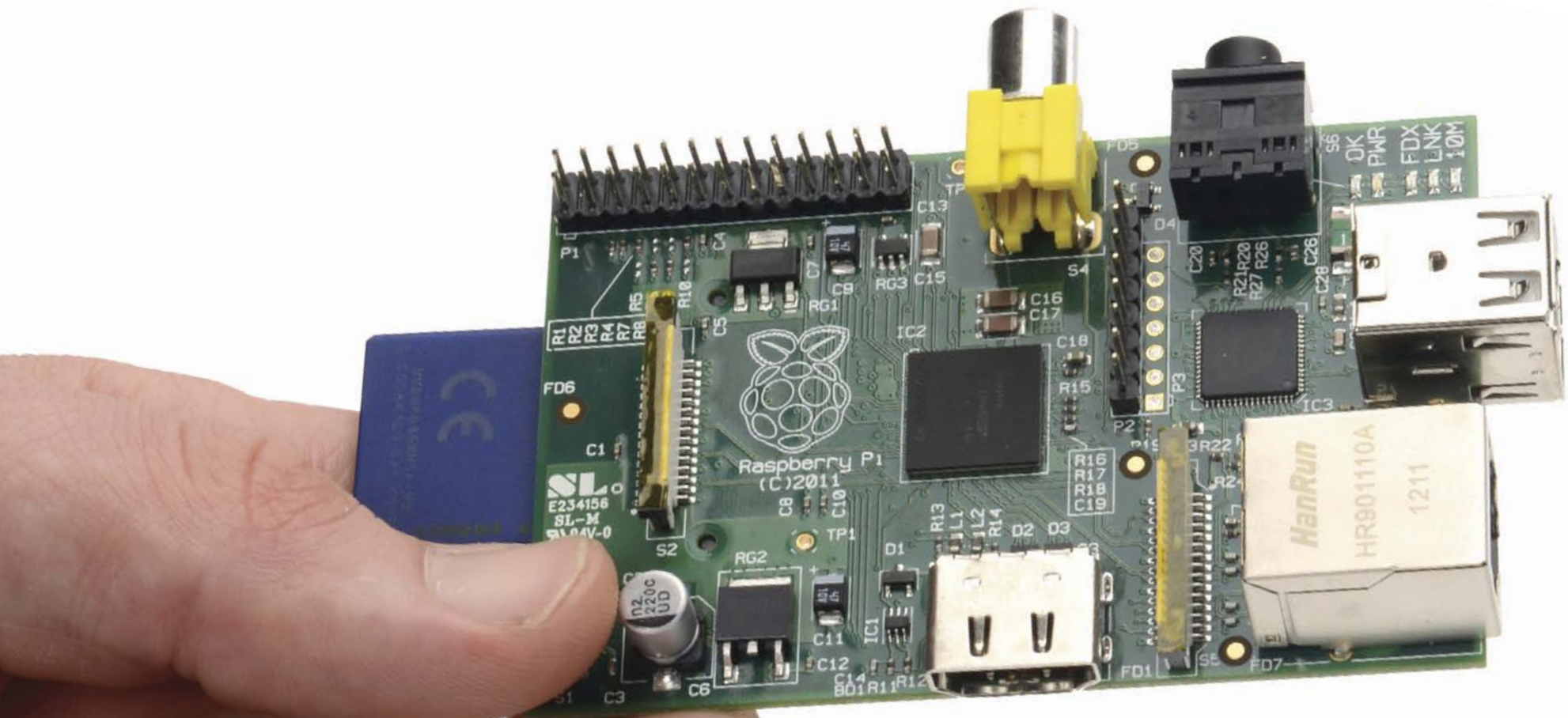
```
def prepare2d(width, height):
    glViewport(0, 0, width, height)
    glMatrixMode(GL_PROJECTION)
    glLoadIdentity()
    glOrtho(0.0, width, 0.0, height, 0.0, 1.0)
    glMatrixMode(GL_MODELVIEW)
    glLoadIdentity()
```

Digging into the details of what these functions are and why you need them would probably fill another full article. For now, consider these functions as further reading for the student.

Talking Pi

Join the conversation at...

@linuxusermag | Linux User & Developer | @ RasPi@imagine-publishing.co.uk

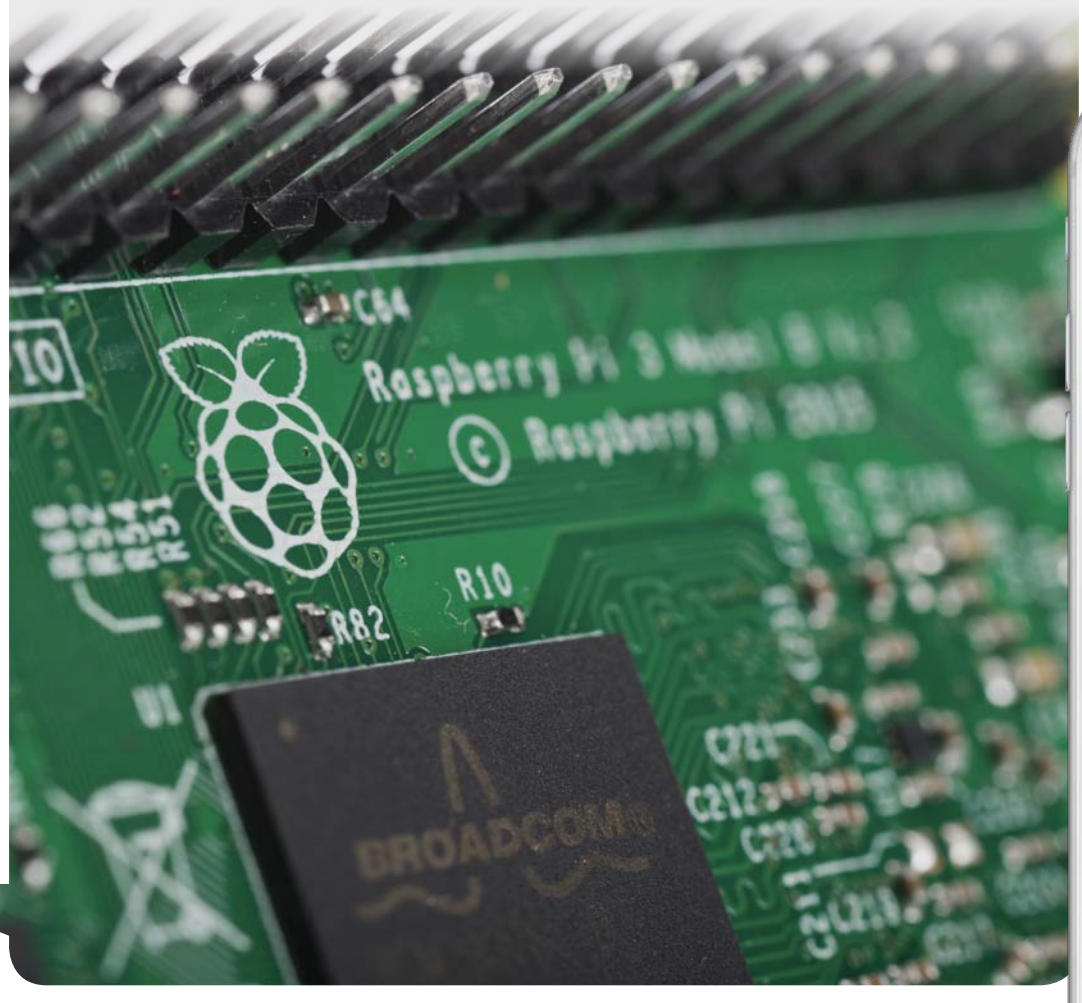


The Raspberry Pi encourages a lot of hands-on work and this means it doesn't quite work like – or as easily as – your laser focus-tested smartphone interface, or even a normal computer. So we're answering your burning Raspberry Pi questions each issue – get in touch with us on Twitter, Facebook or by email.

How can I easily and securely share data between my Pi and my NAS box? I have the latest Synology one.
Jake via email

Lucky for you Jake; you have a recent Synology NAS, which means that easy and secure data shares are just a few clicks away! If you had any other kind of NAS you'd need to run NFS or a Samba share, both of which can be longwinded to set up correctly, and the former might not be as secure as you'd like.

But you have access to Synology's Cloud Station Drive, which works under Linux, so if you're running Raspbian you should have no problems at all. Simply head over to the Synology website at **https://www.synology.com/en-uk/support/download**, choose your device type from the drop-down list, grab the software and go!



Keep up with the latest Raspberry Pi news by following @LinuxUserMag on Twitter. Search for the hashtag #RasPiMag

JUST A SCORE
WHAT'S YOUR JUST A SCORE?

Have you heard of Just A Score? It's a new, completely free app that gives you all the latest review scores. You can score anything in the world, like and share scores, follow scorers for your favourite topics and much more. And it's really good fun!

Why is my Pi showing less memory than it actually has?

Clare via email

When you run the **free** or **top** commands you may expect to see the amount of memory that's specified on the packaging your Pi came in – 256MB, 512MB or a gigabyte of RAM. But you won't. The reason for this is that some of the memory in your Pi is being used to run its GPU (the graphics processor). While your Pi does indeed have the amount of memory that's specified, some of this is constantly in use when your Pi is up and running. That's why there's a difference.

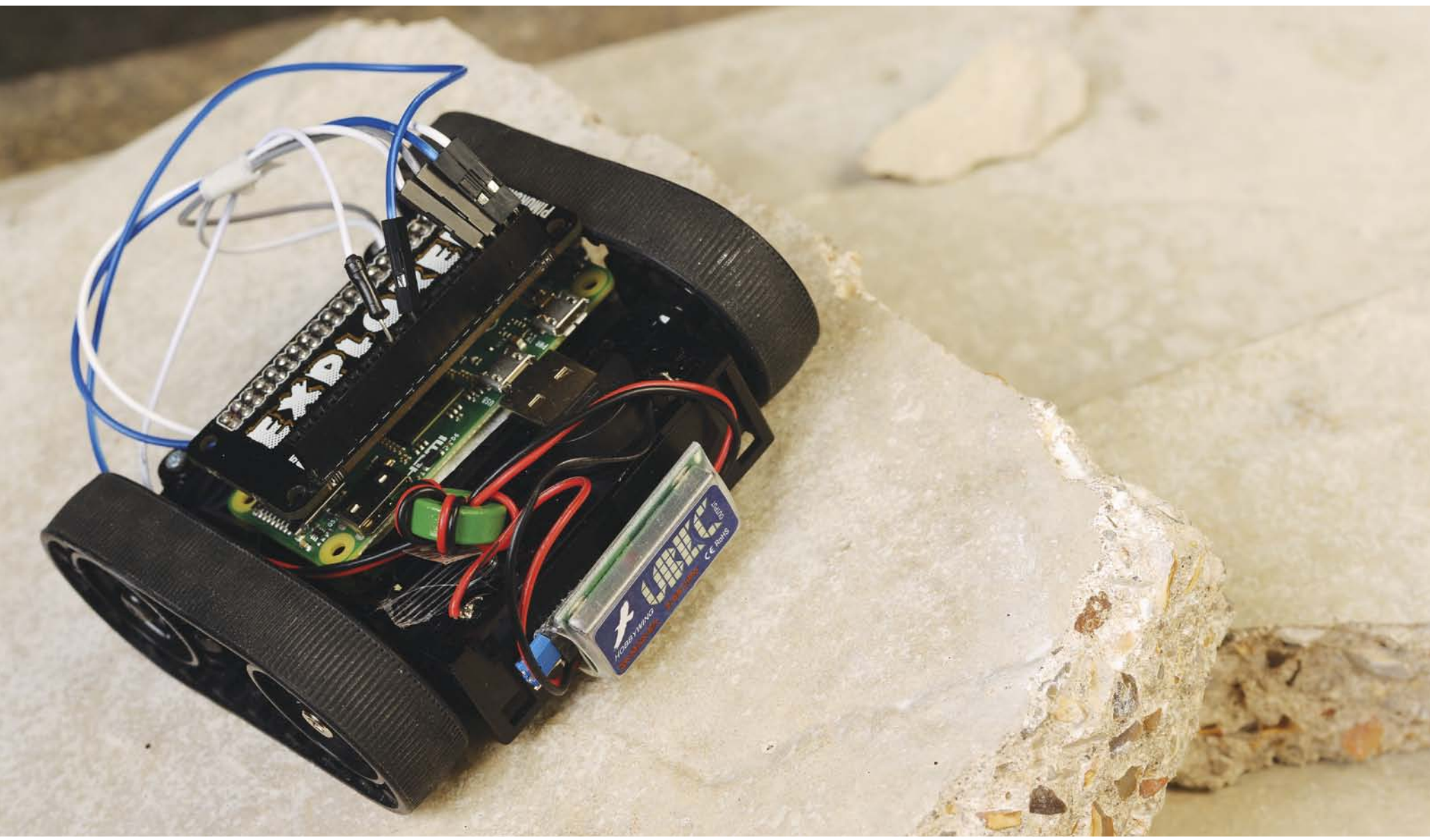
Why doesn't the Pi have a clock like a regular computer does?

Brent via email

According to the Raspberry Pi Foundation, adding a real-time clock (the kind that still keeps its time/date even when the unit is off) is actually prohibitively expensive compared to the rest of the Pi, and would have pushed the price up to unacceptable levels. Anything that keeps the Pi affordable to a wide range of people is fine by us! The Pi Foundation also says that the Pi should happily update its time/date whenever it's connected to a network, but if you're desperate for an always-on chronological device, it's apparently possible to add one yourself using the GPIO pins.

Next issue

Get inspired **Expert advice** **Easy-to-follow guides**



Start building an explorer robot

Get this issue's source code at:
www.linuxuser.co.uk/raspicode